Use case scenario - Flux map integration

X-ray emission, single time slice:

```
>>> te = nstx.s141398.mpts.te(time=0.45)    single Te profile at t=0.45
>>> ne = nstx.s141398.mpts.ne(time=0.45)    single ne profile at t=0.45
>>> sxr = nstx.s0.usxr.generate_xray_profile(te, ne)   use built in method to generate sxr
                                                       profile from Te and ne profiles
or
>>> sxr = nstx.s141398.usxr.generate_xray_profile(time=0.45) method uses mpts profiles directly

>>> sxr2d = nstx.s141398.efit(time=0.45).flux_mapping(sxr) efit method can map 1D to flux surface

>>> sxr_line_integral = sxr2d.line_integrate(nstx.s0.usxr.hup.geometry) diagnostics can have
geometry information, used to find and integrate over sightlines, e.g. 16 USXR HUP channels
```

Entire shot:

```
>>> sxr = nstx.s141398.usxr.generate_xray_profile()    if time not specified, uses all mpts times

>>> sxr2d = nstx.s141398.efit.flux_mapping(sxr)  efit extracts times and maps to each equilibrium

>>> sxr_line_integral = sxr2d.line_integrate(nstx.s0.usxr.hup.geometry) sightline integration for
the entire shot
```

Density line integral:

```
>>> ne2d = nstx.s141398.efit.flux_mapping(nstx.s141398.mpts.ne) 2D mapping of the ne profile

>>> ne_line_integral = ne2d.line_integrate(nstx.s0.firetip.geometry) use FIReTIP geometry to
compare MTPS line integration to FIReTIP data
```

Possible ways to contribute

Users and testers:

- Use FDF data access in your python code
- Provide user feedback for ease of use, bugs, feature requests

Diagnostic support:

- Help write XML module files for diagnostics
- Provide pre/post-processing routines, geometry information where relevant
- Provide specialized analysis/plotting methods

Python developers:

- Develop GUIs
- Write data access/analysis applications (FDFScope)
- Port IDL routines when necessary
- Develop core FDF modules

FDF Development Timeline (Order of Magnitude)

Pre-alpha meeting – end Sept. 2015 (beginning of Oct)
- Demonstrate pre-alpha data framework
- Identify core group of developers/contributors (4-6 people)
- Solicit features & capabilities feedback for initial release

Alpha v0.1 software release – end CY15
- Provide core group with alpha data framework
- Continue feature development with core programming group

MDSPlus / OMFIT coordination – early CY16
- Contact MDSPlus and OMFIT development groups for external coordination
- Identify areas of redundancy or for enhanced synergy
- Solicit suggestions for integration/interoperability with MDSPlus/OMFIT

Beta v0.9 software release – mid CY16
- Feature complete data framework released to NSTX-U research group
- Archival features compatible with NSTX-U Data Management Plan requirements
- Develop IDL / Matlab interface to access Python data framework
- Continue development of data framework applications for data analysis/visualization

Data Framework v1.0 release – late/end CY16
- Comprehensive access to NSTX-U diagnostics, EFIT, TRANSP, other data sources
- Ongoing development of analysis and visualization applications